Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 119 (2015) 486 – 495

Procedia Engineering

www.elsevier.com/locate/procedia

13th Computer Control for Water Industry Conference, CCWI 2015

A tool for practical simplification of water networks models

Daniel Paluszczyszyn[a,*], Piotr Skworcow[a], Bogumil Ulanicki[a]

[a]*Water Software Systems, De Montfort University, The Gateway, Leicester, LE1 9BH, UK*

Abstract

This paper presents development of water network model reduction software, Simplifier2. The application can be integrated with other concepts applied to water distribution system or it can be used as a standalone tool for the purpose of the model simplification only. The utilisation of parallel programming techniques and sparse matrices ordering algorithms drastically increased the speed of simplification. Simplifier2 is able to reduce the water network model, consisting of several thousand elements, in less than 1 minute calculation time. Simplifier2 has been already successfully utilised in a number of research and commercial projects.

Peer-review under responsibility of the Scientific Committee of CCWI 2015

Keywords: simplification; tool; water distribution systems;

1. Introduction

Nowadays it is common that models of water distribution system (WDS) can consist of thousands of elements to accurate replicate the hydraulic behaviour of a real WDS. This approach is appropriate for simulation purposes, however online optimisation tasks are much more computationally demanding and simplified models are needed. [39] proposed a mathematical method for the reduction of WDS models described by a large-scale system of non-linear differential algebraic equations. This procedure has an advantage compared to other methods because the reduced model preserves the nonlinearity of the original network and approximates its operation accurately under different conditions. The method was recently extended in [22] and incorporates the energy audits concepts [6] in order to preserve the energy distribution of the WDS model. The simplified model resembles the energy distribution of original model due to imposing new pressure constraints on the retained consumption nodes. This will prevent a situation where the pump speed required to satisfy minimum pressure constrains is different for the reduced model and the original.

This papers presents development of the model reduction software, Simplifier2, based on the extended simplification algorithm. Simplifier2 could be integrated with other concepts applied to the WDS or it can be used as a standalone tool for the purpose of the model simplification only. Simplifier2 has been already successfully utilised in a number of research and commercial projects, see e.g. [36,37].

* Corresponding author. Tel.: +44-116-207-8939.
E-mail address: paluszcol@dmu.ac.uk

Peer-review under responsibility of the Scientific Committee of CCWI 2015
doi:10.1016/j.proeng.2015.08.871

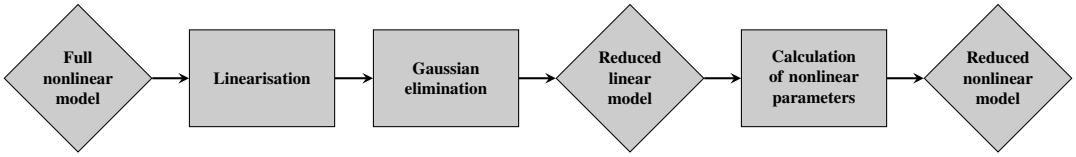


Fig. 1. The variable elimination algorithm.

Section 2 provides a brief description of techniques for WDS model simplification, in particular the variables elimination method. Section 3 gathers all the tools utilised to carry out the implementation. Section 4 outlines the implementation process. Section 5 focuses on the computational aspects arisen throughout the software development. Section 6 briefly describes the features of Simplifier2. Finally, Section 7 concludes this paper.

2. Simplification of water networks models

There are different techniques of a WDS model reduction; the outcome of most of these methods is a hydraulic model with a smaller number of components than the prototype. The main aim of the reduced model is to preserve the nonlinearity of the original network and approximate its operation accurately under different conditions. The accuracy of the simplification depends on the model complexity, purpose of simplification and the selected method such as skeletonization [40], parameter-fitting [2], graph decomposition [9], enhanced global gradient algorithm (EGGA) [16], metamodelling [5] and variables elimination [22,39]. A systematic review of the aforementioned techniques, conducted by [21], recommended the variables elimination as fast, practical and robust technique for simplification of water network models.

2.1. Variable elimination algorithm

The variables elimination method is based on a mathematical formalism initially proposed by [39] and recently extended by [22]. This mathematical method enables reduction of water network models described by a large-scale system of nonlinear differential algebraic equations. The algorithm is illustrated in Figure 1 and proceeds through the following steps: full nonlinear model formulation, model linearisation at specified operation time, linear model reduction using Gaussian elimination and nonlinear reduced model reconstruction. The detailed description of the algorithm is omitted here as it can be found in [1,21]. The approach was successfully implemented and tested on many water networks, e.g. [4,24–26,28–30,35–37]. However, most of the implementations were suited for the particular application, hence no software tool exists that can be either embedded into a larger scheme or used standalone. The motivation behind the work described in this paper, was to create a tool that can be used by water distribution systems community for many purposes. The process of development and numerical enhancements to the method are given in the next sections.

3. Tools and software employed

Development of Simplifier2 was carried out with utilisation of the Microsoft Visual Studio 2010 package. Visual Studio 2010 comes with an integrated support for the .NET 4.0 framework, which enhanced the parallel programming by providing a new runtime, new class library types and new diagnostic tools [20]. These features allowed implementation of the scalable parallel C# code without having to work directly with threads or a thread pool, thereby provided means for improving the performance of numerical calculations.

The input data for the model reduction algorithm are water network topology and simulated hydraulic behaviour of the considered water distribution network. For this purpose the open-source Epanet2 Toolkit [32] was used as a hydraulic simulator to perform an extended period simulation of WDS hydraulic behaviour. The library consists of set of procedures that allow to run/stop simulation, modify simulation and network parameters and read/save the simulation data. The Epanet2 Toolkit provided also a compatibility with ".inp" (INP) format as it is a commonly

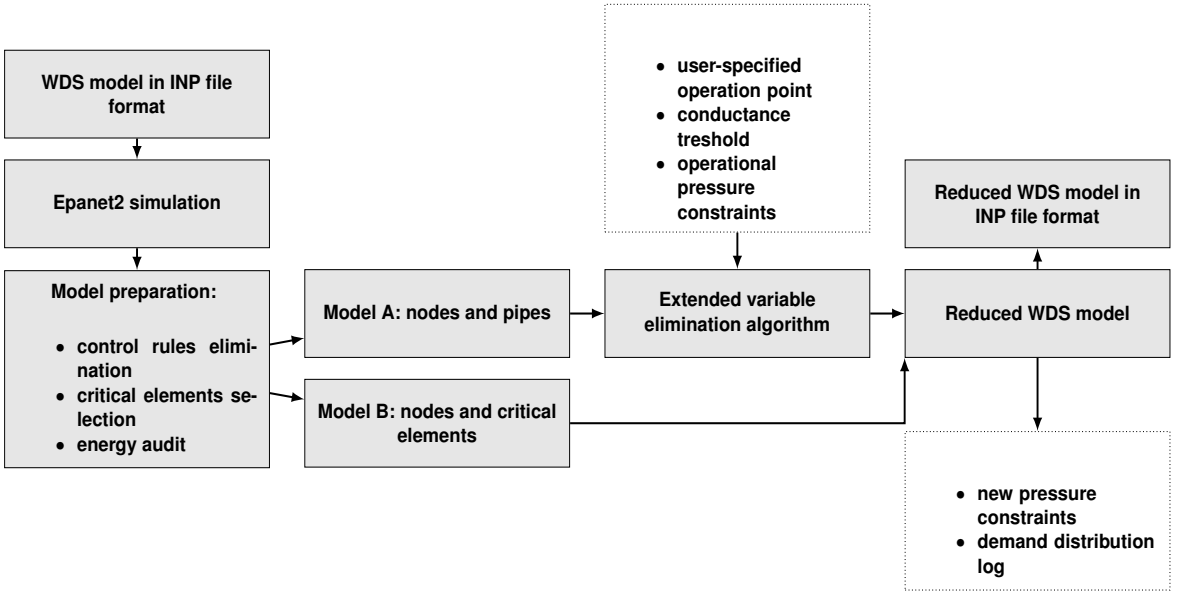


Fig. 2. The overall process of the extended model reduction algorithm.

recognized file format used to store water network models. Unfortunately functionalities of this library are limited and a number of additional C# scripts were written to enable a dynamical hydraulic data export.

A workstation powered by the six-core Intel® Core^{TM} i7 980X processor and 4GB of RAM was used to test performance of the developed tool.

4. Model reduction process

The overview of the overall model reduction process is illustrated in Figure 2. At first, a water network model stored in the INP file format is simulated with the aid of Epanet2 Toolkit to obtain the hydraulic results. Next, the water network model is being inspected to locate any rules or controls associated with the water network elements. Complex and large water networks modelled in Epanet2 often contain rules and controls that can decrease the accuracy of the simplification. It is highly recommended to eliminate the controls and rules, and instead, use time patterns resulting from the simulation of the original model (with control and rules), and associate the patterns with the water network elements. Such an approach serves as a hydraulic benchmark when original and reduced models are compared. Note that in Epanet2 user can associate rules or controls with pipes, transforming them in fact into valves. Since no time patterns can be assigned to the pipe, such rules or controls cannot be automatically eliminated. All components with controls/rules that could not be replaced with a time pattern are automatically selected for retention.

At the preparation stage, the initial energy audit of the considered water network is carried out as described in [6] and [22]. The model preparation stage involves also a selection of other important hydraulic elements to be retained. Initially, it was assumed that operator, based on his knowledge about the particular WDS would choose network elements with a significant importance in order to preserve hydraulic characteristics for wide range of operating conditions. But, even though that this operation needs to be done once for the particular model, it could be a difficult and time consuming task.

Although, a typical hydraulic simulation model contains thousands of pipes but only several tanks, pumps or control valves. Therefore, it is adopted strategy here to reduce the number of pipes and nodes only and retain all other important elements. The identified non-pipe components of a WDS are listed in Table 1. The default is to retain all these elements, but alternatively, the user can define a list of additional elements not to be removed.

To help out the user in a decision-making process, which elements should be additionally retained to replicate more accurate the hydraulic behaviour and layout of original water network, few tools were introduced at the preparation

Table 1. Important elements in water distribution system.

Water distribution system elements
Variable-head reservoirs (tanks)
Forced-head reservoirs
Pumps
Valves
Pipes with associated controls or rules (specific to Epanet2)
Nodes connected to any of the above

stage. They allow selection of nodes, based on their degree; i.e number of neighbours, or select pipes based on their diameter, an approach adapted inter alia by [28]. Nodes with many neighbours are often selected for pressure logger locations. In turn, large diameter pipes often form skeleton of the network. Hence, option to preserve critical nodes and large diameter pipes provides means to retain layout of the network, which is important in WDS design optimisations.

Taking into account the aforementioned considerations, the input model is split up into the two sub-models. Sub-model A, containing pipes and nodes, is subjected to the extended reduction algorithm described in [22] and afterwards, reunited with the other part containing non-pipe and important elements (Sub-model B) to form the complete reduced model, which is saved in the INP file format. Additional output files contain the demand distribution log and new operational pressure constraints.

5. Improving numerical efficiency of simplification algorithm

Once the overall simplification process was implemented, it was subsequently tested on a number of water network models. While the achieved results were satisfactory from the hydraulic perspective, the computational time required to reduce large and medium size networks was in order of hours. Obviously, such long computational time is not desirable, especially for online optimisation strategies. Therefore it was decided to improve the numerical efficiency of simplification algorithm. The following computational aspects of the implementation were investigated in order to reduce the overall time of the model reduction process.

5.1. Parallel programming

Firstly, the focus was placed on the performance of matrix operations. The model reduction algorithm involved a number of matrix multiplications, thereby the speed of these calculations is factor with a profound influence on the total algorithm calculation time.

It was decided to investigate suitability of the model reduction algorithm for a parallel programming in order to exploit the potential of recent multi-core CPUs. The parallel programming is often employed for highly compute-intensive algorithms. It follows the basic idea of decomposition or division of data to be computed asynchronously by each processors. The process of decomposition is dependable on algorithm to be parallelised and type of parallel computing architecture. A number of concurrent programming models were developed over the years e.g. message passing interface (MPI) or multi-threading (see [31] for details). In general, all of them have a static or dynamic period for partitioning or dividing data quantity to be computed in each processor and, eventually, a subsequent utilisation period of intermediate computations to compute the final result.

There is universal agreement that writing multi-threaded code is difficult [38]. Fortunately, .NET 4.0 Framework enhanced the parallel programming by providing a new runtime, new class library types and new diagnostic tools [20]. These features allowed the implementation of the scalable parallel C# code without having to work directly with threads or a thread pool.

Hence, only the most compute-intense and suitable parts of the model reduction algorithm were subjected to parallelisation. This includes calculation of the Jacobian matrix of a considered system and inner loops of the Gaussian elimination process. The inclusion of the parallel programming techniques drastically reduced the algorithm calculation time. Table 2 contains the simplification run-times for a medium size network which contained 3535 nodes,

3279 pipes, 12 tanks, 5 reservoirs, 19 pumps and 418 valves. As can been seen, each successive improvement to the method, reduces the simplification run-time. These enhancements are described in more detail in the following sections.

Table 2. Times taken to complete the simplification process for a medium-sized water network. The benchmark network contained 3535 nodes, 3279 pipes, 12 tanks, 5 reservoirs, 19 pumps and 418 valves. The run-times provided in this table are in the incremental order; the subsequent enhancement uses the predecessor.

Enhancement	Simplification run-time [s]	Reduction of run-time [%]
Initial time	5761	0
Parallel programming *(2 CPU threads)*	4417	23.33
Parallel programming *(4 CPU threads*)	2217	61.52
Parallel programming *(12 CPU threads*)	758	86.84
Matrix storage (*Single-indexed jagged arrays accessed in ijk order*)	95	98.35
Node reordering *(Cuthill-McKee (CMK))*	5	99.91

5.2. Matrix storage

The topology of water network can be represented as an incidence matrix that describes the connectivity between pipes and nodes. Such a representation is also useful for the implementation purposes as the network topology can be explicitly stored in one of the available data structures in the C# language specification.

The considered C# data structures were single-dimensional arrays, multi-dimensional arrays and jagged arrays (arrays of arrays). A single-dimensional array is a list of variables where access to its elements is trough an index. A multi-dimensional array has two or more dimensions, and an individual element is accessed through the combination of two or more indices. A jagged array is an array of arrays in which the length of each array can differ. Jagged array elements are accessed also with two or more indices [34].

One of the techniques often used by programmers to speed up matrix operations is flattening, i.e. representation of multi-dimensional arrays using single-dimensional arrays. Flattening multi-dimensional array into single-dimensional array could benefit in better performance as in the .NET Framework single-dimensional arrays have faster access to their elements, due to optimizations in C# Common Language Runtime (CLR). Hence, the usage of jagged arrays could improve matrix computations as jagged arrays are made of single-dimensional arrays.

While the jagged arrays perform similar to single-dimensional arrays in terms of matrix operations, the jagged arrays were used as they can utilise more effectively the available memory. The maximum-object size in CLR in .NET 4.0 Framework is limited to 2 GB for 32-bit application [19]. Moreover, due to CLR memory overheads, the actual memory limit is around 1.3 GB [3]. Tests performed on the host machine reveal the memory allocation limits for data C# data structures (see Table 3). As can be seen in Table 3, tests with the jagged arrays were able to allocate the

Table 3. Memory allocation limits for C# data structures. Tests were performed on the workstation with 4 GB RAM. Note that in C#, the size of *double* type is 8 bytes.

Data structure	Maximum allocated memory [Megabytes]	Maximum size of $n \times n$ matrix of *double* elements
2D array	1001	11185
Flatten array	1183	12160
Jagged arrays	1530	13829

largest amount of memory. It is because due to memory fragmentation it is easier to find available memory for jagged arrays, i.e. it is more likely that there will be number of blocks of smaller size available than a single, continuous block of the full size of the array, which is required to allocate single and multi-dimensional arrays.

Moreover, the performance of jagged arrays can be improved by using an appropriate order of indices *ijk* in the matrix multiplication algorithm (see [17] for details).

The combination of parallel programming and jagged arrays (single-indexing access, ijk-order) reduced the overall simplification time of the benchmark water network used in Table 2 to 95 seconds, achieving 98.35% decrease with the respect to the initial time.

5.3. Node removal ordering

The Gaussian elimination is the most compute-intensive procedure of the model reduction algorithm. When dense matrices are considered one iteration of the Gaussian elimination uses $O(n^2)$ arithmetic operations and as n iterations must be performed resulting this procedure needs $O(n^3)$ arithmetic operations to complete [18].

Since its introduction, Gaussian elimination and its performance is in a very strong interest for researchers from many disciplines, especially in areas where Gaussian elimination is applied to a sparse matrix. Many variations were developed over the years, often designed for a particular application. The variant of Gaussian elimination used in the model reduction algorithm is given in Algorithm 1. Algorithm 1 has three nested loops with loop indices denoted *k*,*i*,*j*.

Algorithm 1 Gaussian elimination used in the model reduction application.

Require: J, n, n_r ▷ J - Jacobian matrix, n - number of nodes, n_r - number of nodes to remove

```
for k = n to n − n_r do
    if J_kk ≠ 0 then
        for i = 1 to k do
            J_ik ← m_ik = J_ik / J_kk
        end for
        for i = 1 to k do
            for j = 1 to k do
                J_ij = J_ij − m_ik × J_kj
            end for
        end for
    end if
end for
```

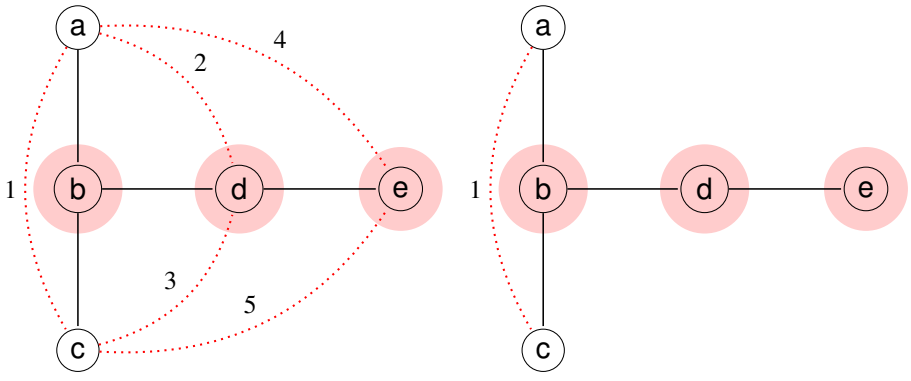
[33] noted that Gaussian elimination on the original matrix results in disastrous fill-ins. Fill-ins are additional non-zeros generated during the elimination. To illustrate this consider a simple network in Figure 3. Nodes *b, d* and *e* are to be deleted from the network. When the process of removal starts from node *b* and then in order *d* and *e*, additional links (indicated by dotted lines) are created between any two nodes that were adjacent to removed node. For *bde* order five links (fill-ins) were created, see Figure 3a. Whereas when starting removal from node *e* and then *d* and *b* only one fill-in (between *a* and *c*) was added, see Figure 3b.

Therefore, the aim of the most researchers is to produce much less fill-ins during Gaussian elimination and thereby reduce computation time and storage space. To address the problem of fill-ins a common-used technique called reordering can be applied to sparse matrices [27]. The idea is to permute the sparse matrix's rows or columns or both. By applying reordering algorithms, the zero and non-zero elements of a sparse matrix are rearranged such that the Gaussian elimination deals with it much more efficiently.

The amount of fill-ins depends on the chosen ordering [27]. Because the fill-in minimisation is impossible to solve in practice heuristics are used [8]. The most widely recognised and applied ordering algorithms are Cuthill-McKee (CMK) [7], reversed Cuthill-McKee (RCMK) [13], minimum degree (MD) [12], Gibbs-Pool-Stockmeyer (GPS) [15] and nested dissection (ND) [14].

Nevertheless, following [13] observations that reversed Cuthill-McKee ordering yields a better scheme for sparse Gaussian elimination it was decided to incorporate RCMK to reorder Jacobian matrix prior Gaussian elimination. The original RCMK algorithm goal is to order nodes locally so that the adjacent nodes are ordered as close as possible.

Note that Gaussian elimination, seen in Algorithm 1, is applied from the bottom of the Jacobian matrix, hence the obtained RCMK ordering was reversed accordingly (RCMK becomes CMK).



(a) Order of removal: *bde*. #fill-ins = 5 (b) Order of removal: *edb*. #fill-ins = 1

Fig. 3. Change in the number of generated fill-ins (additional links indicated by the dotted lines) due to the order of nodes removal.

The CMK algorithm used to reorder nodes prior calculation of Jacobian matrix is given in Algorithm 2 [33]. Its queue-based implementation was adapted for water network model reduction i.e. only nodes to be removed are ordered. The effectiveness of CMK algorithm depends critically on the choice of starting node. The starting node may be one of minimum degree [27] or pseudo-peripheral node as proposed by [10]. Here, the latter heuristic was implemented to determine the best starting node for CMK ordering.

Algorithm 2 Cuthill-McKee ordering of nodes prior to Gaussian elimination

Require: V ▷ Set of all nodes
$K \subset V$ ▷ Subset of nodes to be kept
$R \subset V$ ▷ Subset nodes to be removed
$Find \quad n_0 \in R$ ▷ Find starting node n_0
$Q = \{n_0\}$ ▷ Queue of nodes
$\pi = \varnothing$ ▷ Set of ordered nodes
while $NotEmpty(Q)$ **do**
  $node = Q.Dequeue$
  if $node \notin \pi$ **then**
    $\pi.Append(node)$
    $Sort(node.neighbours)$ ▷ Sort all node neighbours based on their degree
    for all $neighbours$ **do**
      if $neighbour \notin \pi$ **then**
        $Q.Enqueue(neighbour)$
      end if
    end for
  end if
end while
return $K \cup \pi$

Although, the post-elimination matrices have a completely different structure there is no difference between the reduced networks. The obtained numerical results were the same in the parts of "reduced" Jacobian matrices that will be used in the next step to recreate the simplified nonlinear water networks.

Ultimately, the choice whether to use of ordering algorithm is determined by size of the water network to be simplified. For water network models with the number of nodes $n < 500$ no ordering is applied. For larger problems the CMK ordering is chosen. Time complexity of CMK for a dense matrix is $O(q_{max}m)$ where q_{max} is the maximum degree of any node and m is the number of links (edges) [11]. However, for sparse matrices the CMK time complexity is reduced to $O(n)$ [23]. The biggest profit from the ordering was reduction of time needed for Gaussian elimination.

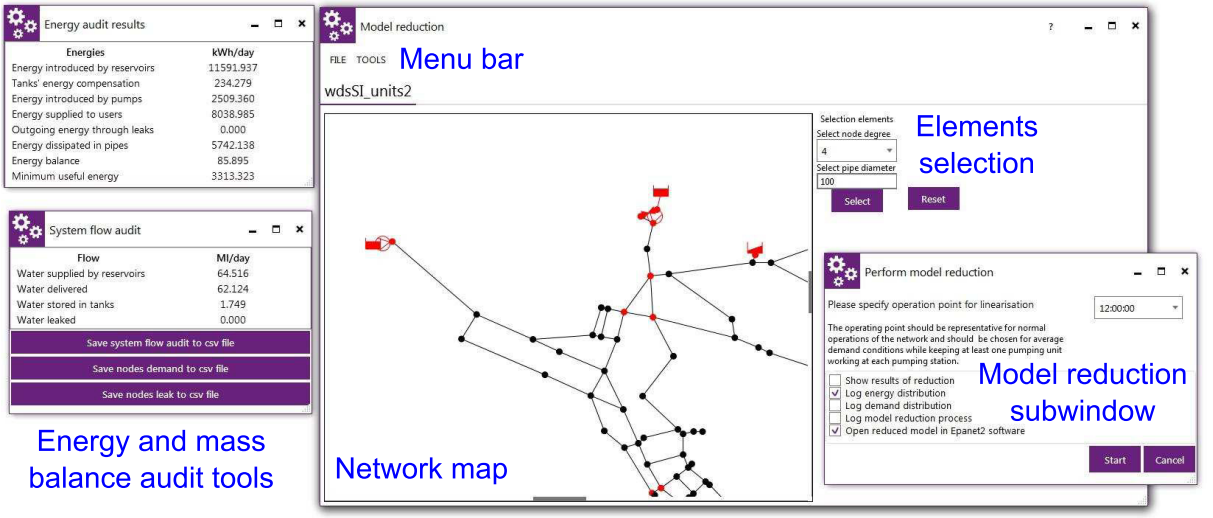


Fig. 4. Illustrating the main window of Simplifier2.

When the model reduction algorithm was applied to the benchmark network used in Table 2, but preceded with the CMK ordering of the Jacobian matrix, the computational time was reduced to less than 5 seconds.

6. Simplifier2

The simplification process, illustrated in Figure 2, evolved throughout its implementation into a more sophisticated and extended tool. The final implementation took into account the outcomes from investigation of parallel programming, storage structures for sparse matrices and nodes pre-ordering.

Simplifier2 was developed to facilitate the extended simplification process of water distribution networks described in [22]. The present tool aims at returning a simplified WDS topology which can be still used to perform hydraulic simulation. The application can be used either as a standalone application or as an embedded module in other applications. Indeed, it forms a key module of Finesse 2 software, the successor of Finesses software (description can be found in [30]), currently being developed by the Water Software Systems (WSS) members.

The main user workspace of Simplifier2 is pictured in Figure 4. The workspace includes the following elements: network map window, menu bar, status bar and water network elements selection toolbox. A concise description of main elements is provided in the next paragraphs.

The menu bar located at the top of workspace contains a collection of menus used to control the application. It provides standard commands for opening, closing, printing and setting application preferences. The menu bar includes also commands to launch tools such as water network energy audit, network system flow and scaling of total network demand.

The network map window provides means to display a schematic diagram of the objects comprising a water distribution network. The displayed topology is created upon data read from the corresponding INP file. The crucial elements (see Table 1) selected to retain are displayed by using different colors. Additionally, the existing objects can be clicked on for marking/unmarking. The map can be printed, zoomed and panned from one position to another. Nodes and links can be drawn at different sizes with ID labels and numerical property values displayed.

The elements selection toolbox is located on the panel right to the network map window. It provides features to mark nodes based on their degree and/or mark pipes based on diameter ranges. Note that all elements listed in Table 1 are marked automatically and cannot be unmarked.

The model reduction sub-window allows selection of the operating point for linearisation and provides means to log the simplification process and demand redistribution, save new pressure constraints and open the simplified model in Epanet2.

Since its development Simplifier2 has been used to reduce many WDS models and has proven to be a practical and reliable tool. An example of utilisation is given in [36], where it has been used in a practical project focused on determining optimal schedules for control elements in a real large-scale water network exhibiting highly complex topology.

7. Conclusions

This paper has dealt with the implementation and further improvement of the extended model reduction algorithm elaborated in [22]. The process of design and development of the research software has been presented with the focus places on the emerged computational research aspects.

Different implementation approaches and their limitations have been investigated. The implementation and graphical user interface were coded in the C# programming language. The Epanet2 Toolkit has been used as a hydraulic simulator to perform an extended period simulation of WDS hydraulic behaviour. Parallel programming techniques have been employed to distribute workload of the algorithm across multiple CPU cores which nowadays are present in majority of PCs. The limitations of available data structures to store the matrix representation of water networks along with the benefits of sparse matrix reordering prior Gaussian elimination have been examined. The utilisation of parallel programming techniques and the sparse matrices ordering algorithms have drastically increased the speed of the model simplification.

There are algorithms that work well for theoretical and small systems, but they often do not consider practical constraints, hence, they are not actually suitable for real systems, whereas in this work a practical tool was created. The developed software is able to simplify the water network model, consisted of several thousands elements, within seconds of calculation time. The advantage of this near real-time model reduction is that can be used to manage abnormal situations and structural changes in a water network, e.g. isolation of part of the network due to a pipe burst. In such case an operator can change the full hydraulic model and run model reduction software to automatically produce the updated simplified model.

Simplifier2 can be integrated with other concepts applied to the WDSs or it can be used as a standalone tool for the purpose of the model simplification only. The present tool aims at returning a simplified WDS topology, which can be still used to perform hydraulic simulation. Although, not necessarily relevant for the simplification process, the features of energy balance audit, system flow audit and scaling of total system demand may prove to be useful and applicable for other research purposes.

References

[1] Alzamora, F., Ulanicki, B., and Salomons, E. (2014). A fast and practical method for model reduction of large-scale water distribution networks. *Journal of Water Resources Planning and Management*, 140:444–456.

[2] Anderson, E. and Al-Jamal, K. (1995). Hydraulic-network simplification. *Journal of Water Resources Planning and Management*, 121(3):235–240.

[3] Ayucar, I. (2012). Memory limits in a .NET process. http://www.codeproject.com/Articles/483475/Memory-Limits-in-a-NET-Process. Accessed on May 2014.

[4] Bounds, P., Kahler, J., and Ulanicki, B. (2006). Efficient energy management of a large-scale water supply system. *Civil Engineering and Environmental Systems*, 23(3):209 – 220.

[5] Broad, D., Maier, H., and Dandy, G. (2010). Optimal operation of complex water distribution systems using metamodels. *Journal of Water Resources Planning and Management*, 136(4):433–443.

[6] Cabrera, E., Pardo, M., Cobacho, R., and E. Cabrera, J. (2010). Energy audit of water networks. *Journal of Water Resources Planning and Management*, 136(6):669–677.

[7] Cuthill, E. and McKee, J. (1969). Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 24th National Conference ACM '69*, pages 157–172, New York, NY, USA. ACM.

[8] Davis, T. A. (2006). *Direct methods for sparse linear systems*. Siam.

[9] Deuerlein, J. (2008). Decomposition model of a general water supply network graph. *Journal of Hydraulic Engineering*, 134(6):822–832.

[10] George, A., Liu, J., and Ng, E. (1994). *Computer Solution of Sparse Linear Systems*. Academic Press, Orlando, Florida.

[11] George, A. and Liu, J. W. (1981). *Computer Solution of Large Sparse Positive Definite*. Prentice Hall Professional Technical Reference.

[12] George, A. and Liu, J. W. (1989). The evolution of the minimum degree ordering algorithm. *Siam review*, 31(1):1–19.

[13] George, J. A. (1971). *Computer implementation of the finite element method*. PhD thesis, Departement of Computer Science, Stanford University.

[14] George, J. A. (1973). Nested dissection of a regular finite element mesh. *SIAM Journal on Numerical Analysis*, 10(2):345–363.

[15] Gibbs, N. E., Poole, Jr, W. G., and Stockmeyer, P. K. (1976). An algorithm for reducing the bandwidth and profile of a sparse matrix. *SIAM Journal on Numerical Analysis*, 13(2):236–250.
[16] Giustolisi, O. and Todini, E. (2009). Pipe hydraulic resistance correction in WDN analysis. *Urban Water Journal*, 6(1):39–52.
[17] Golub, G. H. and Van Loan, C. F. (2012). *Matrix computations*, volume 3. JHU Press.
[18] Lovász, L. and Gács, P. (1999). Complexity of algorithms. *Lecture Notes, Boston University, Yale University*.
[19] Microsoft (2015a). Memory limits for windows releases. http://msdn.microsoft.com/en-us/library/aa366778 Accessed on May 2015.
[20] Microsoft (2015b). Parallel programming in the .NET framework. http://msdn.microsoft.com/en-us/library/dd460693.aspx. Accessed on May 2015.
[21] Paluszczyszyn, D. (2015). *Advanced modelling and simulation of water distribution systems with discontinuous control elements*. PhD thesis, De Montfort University.
[22] Paluszczyszyn, D., Skworcow, P., and Ulanicki, B. (2013). Online simplification of water distribution network models for optimal scheduling. *Journal of Hydroinformatics*, 15(3):652–665.
[23] Pedroche, F., Rebollo, M., Carrascosa, C., and Palomares, A. (2012). L-RCM: a method to detect connected components in undirected graphs by using the Laplacian matrix and the RCM algorithm. *arXiv preprint arXiv:1206.5726*.
[24] Perelman, L., Maslia, M. L., Ostfeld, A., and Sautner, J. B. (2008). Using aggregation/skeletonization network models for water quality simulations in epidemiologic studies. *Journal of American Water Works Association*, 100(6):122–133.
[25] Perelman, L. and Ostfeld, A. (2006). Aggregation of water distribution systems for contamination detection. In *Water Distribution Systems Analysis Symposium*, pages 1–13, Cincinnati, Ohio, USA.
[26] Perelman, L. and Ostfeld, A. (2008). Water distribution system aggregation for water quality analysis. *Journal of Water Resources Planning and Management*, 134(3):303–309.
[27] Pissanetzky, S. (1984). *Sparse matrix technology*. Academic Press London.
[28] Preis, A., Whittle, A., Ostfeld, A., and Perelman, L. (2009). On-line hydraulic state estimation in urban water networks using reduced models. In Boxall, J. and Maksimović, C., editors, *Proceedings of the 10th International Conference on Computing and Control for the Water Industry (CCWI2009)*, pages 319–324, Sheffield, UK. CRC Press.
[29] Preis, A., Whittle, A., Ostfeld, A., and Perelman, L. (2011). Efficient hydraulic state estimation technique using reduced models of urban water networks. *Journal of Water Resources Planning and Management*, 137(4):343–351.
[30] Rance, J., Coulbeck, B., Kosov, S., Bounds, P., and Ulanicki, B. (2001). FINESSE - a comprehensive software package for water network modelling and optimisation. In *Water Software Systems: Theory and Applications*. Research Studies Press LTD, Baldock, England.
[31] Rauber, T. and Rnger, G. (2010). Parallel programming models. In *Parallel Programming*, pages 93–149. Springer Berlin Heidelberg.
[32] Rossman, L. (2000). *EPANET 2 Programmers Toolkit*. US: Risk Reduction Engineering Laboratory, Office of Research & Development, US Enviromental Protection Agency, Cincinnati, Ohio, USA.
[33] Saad, Y. (2003). *Iterative methods for sparse linear systems*. Siam.
[34] Schildt, H. (2010). *C# 4.0: The Complete Reference*. McGraw-Hill.
[35] Shamir, U. and Salomons, E. (2008). Optimal real-time operation of urban water distribution systems using reduced models. *Journal of Water Resources Planning and Management*, 134(2):181–185.
[36] Skworcow, P., Paluszczyszyn, D., and Ulanicki, B. (2014). Pump schedules optimisation with pressure aspects in complex large-scale water distribution systems. *Drinking Water Engineering and Science Discussions*, 7(1):121–149.
[37] Skworcow, P., Ulanicki, B., AbdelMeguid, H., and Paluszczyszyn, D. (2010). Model predictive control for energy and leakage management in water distribution systems. In *Proceedings of the UKACC 8th International Conference on Control*, pages 990–995, Coventry, UK.
[38] Sutter, H. and Larus, J. (2005). Software and the concurrency revolution. *ACM Queue*, 3(7):54–62.
[39] Ulanicki, B., Zehnpfund, A., and Martinez, F. (1996). Simplification of water network models. In Muller, A., editor, *Hydroinformatics 1996: Proceedings of the 2nd International Conference on Hydroinformatics*, volume 2, pages 493–500, Zurich, Switzerland. ETH, International Association for Hydraulic Research.
[40] Walski, T., Chase, D., Savić, D., Grayman, W., Beckwith, S., and Koelle, E. (2003). *Advanced water distribution modeling and management*. Haestad Press, Waterbury, CT USA, 1 edition.